Exploring Pole Balancing on a Quadrotor

Quincy Johnson
MIT, Cambridge, United States
qjohnson@mit.edu

***Abstract*—We explore the problem of balancing a pole using a quadrotor. Balancing and swing up controllers serve as a good baseline for analyzing control methods and can be important sub-goals in more complicated tasks such as catching or acrobatic maneuvers. We explore different approaches for controlling balancing, and attempt to address the issue of handling modeling errors that arise from using controllers on systems that were originally meant for controlling a simpler system representation. We use infinite horizon discrete time linear quadratic regulator (LQR) and optimization based model predictive control (MPC) in our analysis.**

I. Introduction

There have been many impressive achievements in robotic acrobatic maneuvers over the pass few years. Researchers/roboticists have explored many different approaches for controlling such systems. Model based control approaches are very powerful for achieving these complicated tasks if the full state dynamics are known and can be accurately modeled. In such cases, one effective method of control is to linearize the system and design an LQR controller on the linear system. For state-spaces near the nominal state, LQR has proven to be quite capable in generating a control policy towards the nominal state and demonstrated to be fast/have good convergence [1]. For the purposes of our task of balancing a pole on a quadrotor, a model based control method seems viable. We will show how the full state dynamics can be derived and that there is a known system equilibrium point. Planning also needs to be fast to react to perturbations in the system that may cause a departure from equilibrium and the pole to fall over. With this, we will then show the limitations of finite horizon discrete time LQR and optimization based MPC.

II. Related Work

The original inspiration for this work came from a dual-quadrotor system that tossed and caught a pole between the two [2]. They formulate the problem by breaking down the system in 3 states: planning the throw trajectory, planning the catch trajectory, and pole balancing when in between the throw/catch states. The pole balancing makes use of an a infinite horizon LQR, and follows a balancing control scheme [3] where the complex system consisting of a free-body pole and quadrotor is simplified, where they make the assumption that the pole is rigidly attached to the center of mass of the quadrotor. This system is then linearized around derived equilibrium states for LQR to produce a policy towards these nominal balanced states to be run on the original full-state system. We take a very similar approach for balancing in this work. Other approaches for balancing poles in non-linear systems attempt to model and be robust to disturbances [4]. In this work we ignore all external forces except gravity that a real system might encounter, such as drag.

III. Methodology

All systems and written code were constructed in the Drake [5] environment.

A. Dynamics

The main "complex" system consists of a floating-base rod and a quadrotor which both states are composed of translational position and velocity, as well as its orientation and angular velocity about it's center of mass, all expressed in the world frame W_o such that both have states of size 12. We'll choose to exclude the angular velocity from the quadrotor state since it isn't directly controllable and but is subject to the dynamics of the system. Therefore the quadrotor state is $q_s = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \alpha, \beta, \gamma)$ where the vehicle frame's orientation is $R_z(\alpha) * R_y(\beta) * R_x(\gamma)$ The quadrotor takes as input the thrust a_i to command propeller p_i to generate for each of the four propellers. To make derivations easier down the line, this control input scheme u=(a_1, a_2, a_3, a_4) is converted to u_q=$(a, \omega_x, \omega_y, \omega_z)$, where a is the mass normalized collective thrust, ω_x, ω_y, and ω_z are the desired rotational rates about the vehicle body axes. The equations to get back the original control input (a_1, a_2, a_3, a_4) from $(a, \omega_x, \omega_y, \omega_z)$ is:

$$a_1 = \frac{1}{4} \times \left(\frac{(a \cdot k_M - \omega_z \cdot k_F) \cdot L_r - 2 \cdot \omega_y \cdot k_M}{k_F \cdot k_M \cdot L_r} \right)$$
$$a_2 = \frac{1}{4} \times \left(\frac{(a \cdot k_M + \omega_z \cdot k_F) \cdot L_r - 2 \cdot \omega_x \cdot k_M}{k_F \cdot k_M \cdot L_r} \right)$$
$$a_3 = \frac{1}{4} \times \left(\frac{(a \cdot k_M - \omega_z \cdot k_F) \cdot L_r + 2 \cdot \omega_y \cdot k_M}{k_F \cdot k_M \cdot L_r} \right)$$
$$a_4 = \frac{1}{4} \times \left(\frac{(a \cdot k_M + \omega_z \cdot k_F) \cdot L_r + 2 \cdot \omega_x \cdot k_M}{k_F \cdot k_M \cdot L_r} \right)$$

Where k_M and k_F are the thrust and moment ratio properties of the quadrotor and L_p is the length of the rotor arms. Refer to Figure 1.

The goal is to produce a balancing policy for this system. In order to produce a controller for this more complex system, we first consider a simpler system in which we assume the pole's base is rigidly attached to the center of the vehicle's mass. In this way the problem is reduced to a swing-up/balance inverted pendulum problem. Doing so simplifies the system's

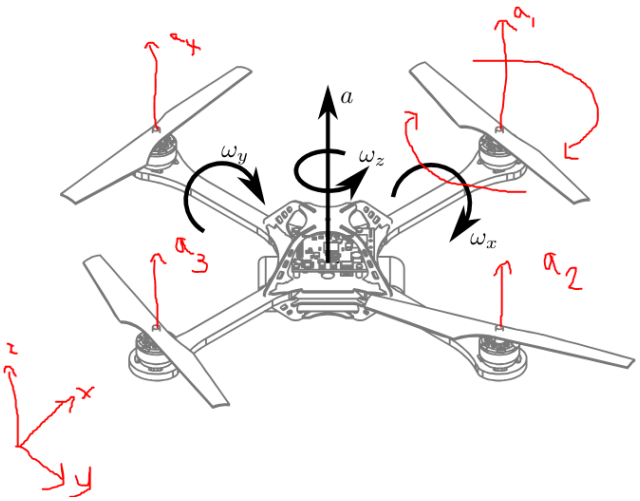


Fig. 1. Quadrotor dynamics adapted from [6]

state such that the pole is no longer a floating base and the state of the pole is now $(\theta, \phi, \dot{\theta}, \dot{\phi})$ where θ and ϕ define how much the pole has rotated over the x and y axis respectively in the vehicle frame. This state can also be expressed as its translational position of the pole's center relative to the vehicle base. (We'll define this to be r along the x-direction of the vehicle and s along the y-direction). Now the state of the quad-pole system, q_{sp} is $(q_s, r, s, \dot{r}, \dot{s})$, and the dynamics is given by $f(q_{sp}, u_q)$. See [3] for the full dynamics derivations.

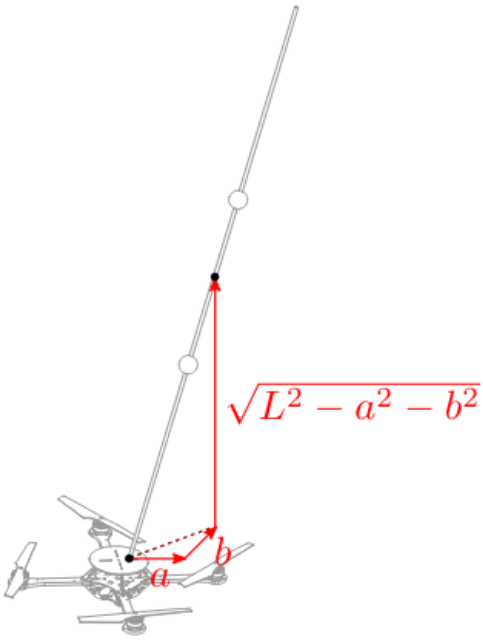


Fig. 2. Simplified pole state [2], where a,b correspond to r,s defined earlier. L = 0.5 * length of pole.

$$\ddot{\tilde{r}} = \tilde{r}\frac{g}{L} - \tilde{\beta}g$$
$$\ddot{\tilde{s}} = \tilde{s}\frac{g}{L} + \tilde{\gamma}g$$
$$\ddot{\tilde{x}} = \tilde{\beta}g$$
$$\ddot{\tilde{y}} = -\tilde{\gamma}g$$
$$\dot{\tilde{\beta}} = \tilde{\omega}_y$$
$$\dot{\tilde{\gamma}} = \tilde{\omega}_x$$
$$\ddot{\tilde{z}} = \tilde{a}$$

Fig. 3. Linearized dynamics around upright-pole equilibrium point [3]

B. Control

To balance the pole in the simple system, we need an equilibrium state-input pair. One intuitive equilibrium point is the pole in a stationary upright position, the quadrotor is at rest around some nominal position, and the only control input is just a collective thrust to counteract gravity:

$$q_{sp0} = (x_0, y_0, z_0, \alpha = \alpha_0, \beta = 0, \gamma = 0, v_x = 0, v_y = 0, v_z = 0, r = 0, s = 0, \dot{r} = 0, \dot{s} = 0)$$

$$u_{q0} = (-(m_p + m_q) * g, 0, 0, 0)$$

Linearizing the system around this equilibrium point yields the following dynamics shown in figure 3. We then use infinite horizon discrete time LQR to produce a policy that achieves this nominal state-input pair for the simplified system.

To produce a controller for the original complex system, we reason that the complex system is close enough to the simplified system when the complex system is initialized to be very close to the nominal state that way the base of the pole is in full contact with the quadrotor. Otherwise, we expect LQR to fail to produce a good policy. To get LQR to work on the complex system we explore two methods. The first has the complex state converted into the same state representation as the simplified system and then is pass to the linearized simplified system to be initialized from the converted state. The control policy from the trajectory generated from applying LQR to this initialized linear system is then input into the original complex system. The second method initializes both the simplified system and the complex system. LQR generates a control policy from the linearized simplified system, but is fed in states from the complex system's converted state at each time step. The produced command is then fed back to the complex system.

Another briefly explored control method for balancing was using direct collocation model predictive control. This method was disbanded as the controller took too long to generate solution trajectories before the pole fell over.

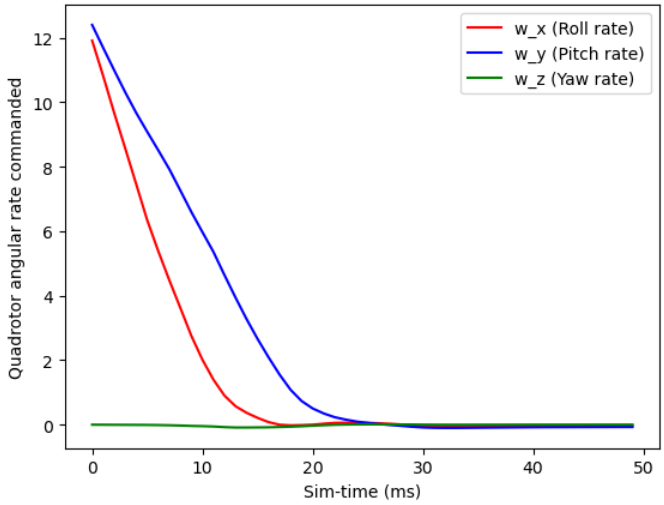


Fig. 4. Commanded angular rates from LQR on the simplified system

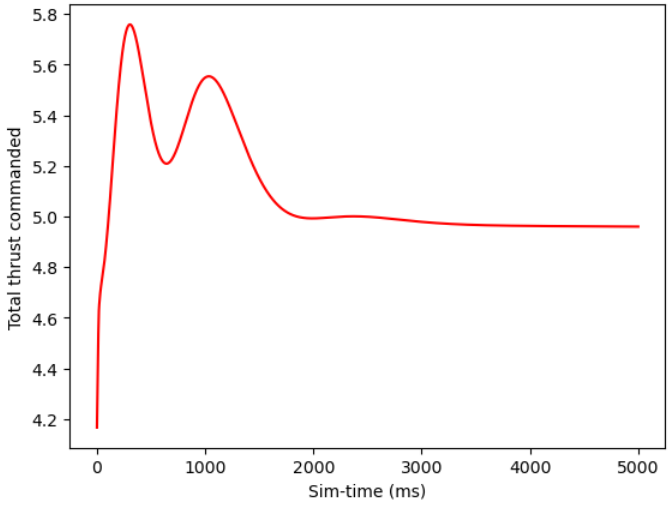


Fig. 6. Commanded thrust from LQR on simplified system. (Nominal is near 5.0)

IV. Results

A. Simplified System

The following figures 4-7 are the results of running LQR on the simplified system averaged over 20-5 second trials from the same distance of 2m away and a starting θ, ϕ pole angle between $\pm\pi/6$ rads. LQR is successfully able to balance the simplified pendulum-esque system and converge to the nominal state fairly quickly.

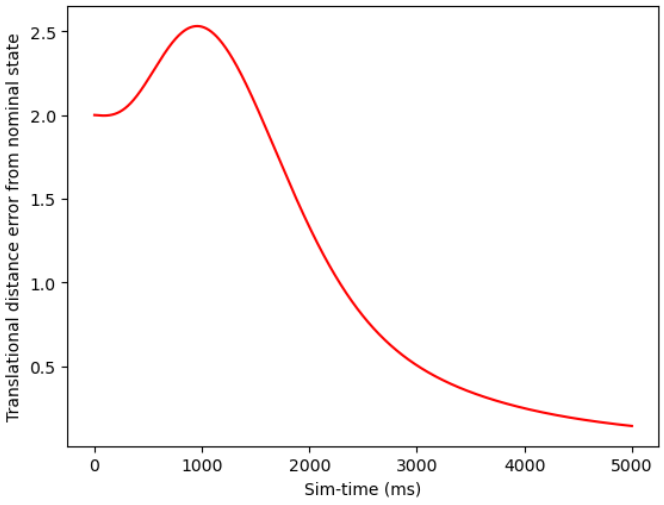


Fig. 5. Quadrotor positional error from LQR running on the simplified system

The Figures 8-9 shows the method of directly feeding in the complex system's converted state into the LQR controller. Converges to the nominal state fails and the system quickly diverges/explodes. This is likely due to the state of the complex system evolving in a non-linear way rather than from what LQR expects, which its policy isn't optimized for, so even deviating a little bit from this expected trajectory

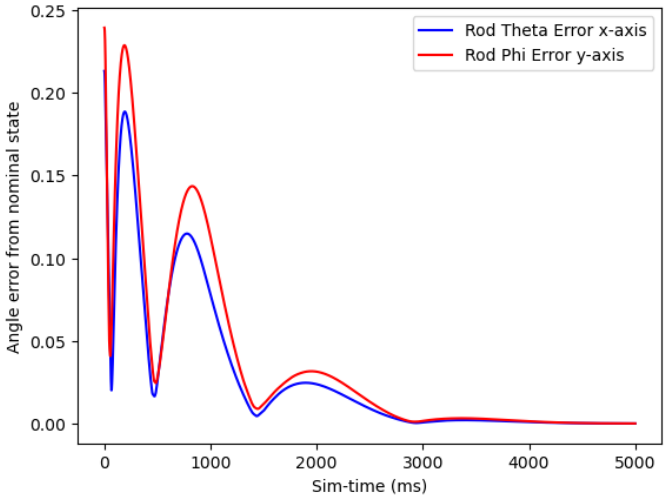


Fig. 7. Pole angle error from LQR running on simplified system

evolution leads to nonsensical control outputs.

Figure 10 shows the results of directly applying the linearized simplified system's LQR policy to control the complex system. The system is able to balance briefly with very minimal angle error just by directly using the LQR policy derived from the linearized simple system. However, the theta and distance error eventually diverge, indicating a fallen pole. This likely happened due to the accumulated differences in the modeled systems of the simple vs the complex system.

V. Discussion

Balancing the pole on the full-state complex system, where the rod is not assumed to be attached at the base to the quadrotor, proved to be a non-trivial task. From the results, it seems that the moments applied to the base of the rod make a large impact on the dynamics of the system, such that the policy

produced by the trajectory generated from running LQR on the simplified system is insufficient for effectively balancing on the real complex system. It would be interesting to explore if longer balancing can happen if the simple system's state is updated periodically to match the complex system's current state and then rerun LQR on the simple system each time this update occurs. That way, we avoid the situation where LQR only outputs nominal commands when the simple system reaches equilibrium, even though the complex system hasn't.

The original goal of the project was to implement pole catching and throwing. Moving forward with the project, if failure to produce a good long long-lasting pole balance strategy using LQR continues, then we will turn our attention to methods that don't involve linearizing the system or directly need the dynamics of the system, such as reinforcement learning. Given enough training, reinforcement learning may be able to produce a fairly robust policy. A linear mpc approach, which takes into account the friction cone between the pole and the quadrotor, could also prove promising.

REFERENCES

[1] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 4.

[2] D. Brescianini, M. Hehn, and R. D'Andrea, "Quadrocopter pole acrobatics," 11 2013, pp. 3472–3479.

[3] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 763–770.

[4] L. B. Prasad, B. Tyagi, and H. O. Gupta, "Optimal control of nonlinear inverted pendulum dynamical system with disturbance input using pid controller lqr," in *2011 IEEE International Conference on Control System, Computing and Engineering*, 2011, pp. 540–545.

[5] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[6] H. Zaki, M. Unel, and Y. Yildiz, "Trajectory control of a quadrotor using a control allocation approach," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 533–539.

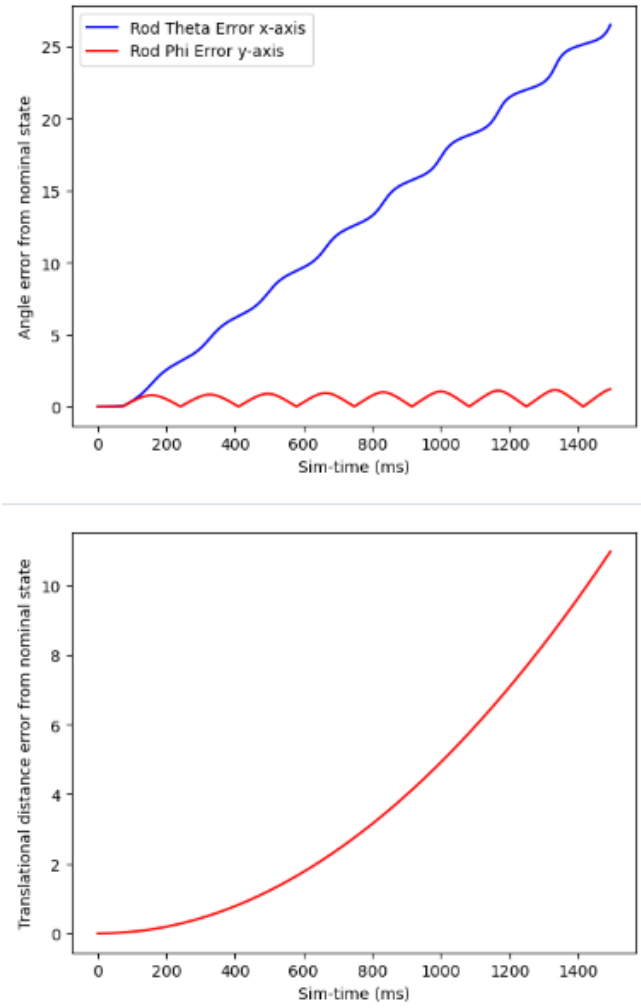
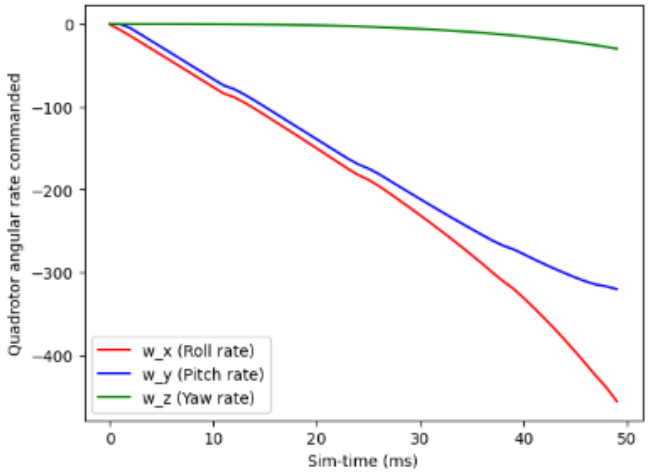


Fig. 8. Pole angle and quadrotor distance error when being fed commands from LQR running on a complex full-state system. System initialized close to nominal state.



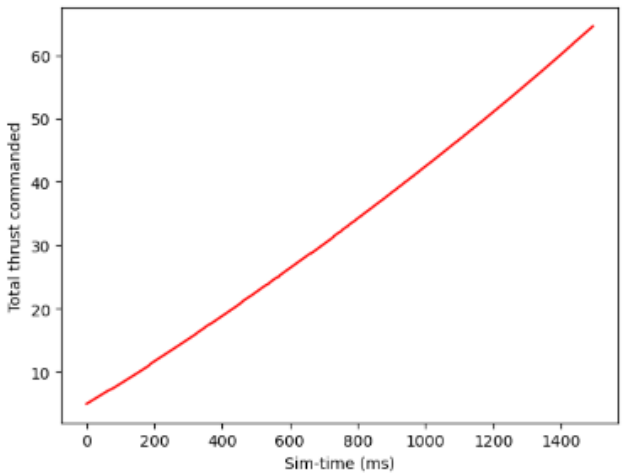
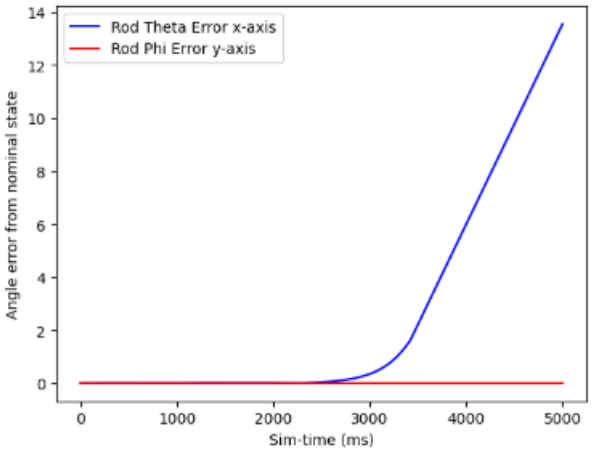


Fig. 9. Commands generated from LQR running on a complex full-state system. System initialized close to nominal state.



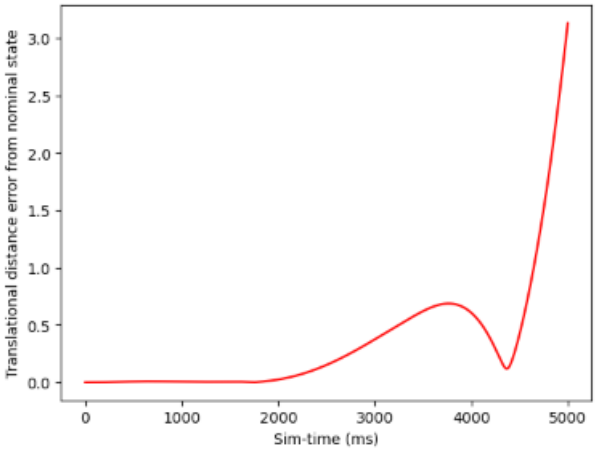


Fig. 10. Pole angle and quadrotor distance error when being fed commands from LQR running on simplified system near nominal state $(\theta, \phi <= 0.1 rads)$.